The Mad Programmer: Paul Holmgren
Igor: Willie Jones

Welcome to a new experience in Timex telecommunications.

The Indiana Sinclair Timex users Group was formed as a result of Frank Davis's work after the Mid-West TS Computer Fest. Several of the members left the Fest with a copy of T/S Tinyboard by Randy Gordon. At an early meeting it was revealed that one of our members had an extra phone line and wanted to run Tinyboard as a service to the membership.

After we got the BBs up and running he found that he could not keep the darn thing running very well. It was decided that the program needed looking at. After many examinations we decided to start over with the goals of designing a reliable and easy to operate BBs.

Tinyboard, in the form we got it, did not work for us. No reflection on Randy, we just needed and wanted more, and set out to get it. After considerable trials and tribulations writing the program, up-loading or Pony Expressing it to the operating location to put it to the test; we succeeded. To test this type of program you really need to use it to gain accurate information when it crashes, of which we had our share.

A strong "Thank You" goes to Willie Jones. He ran each change as they were made available and reported the bugs and crashes accurately, allowing the Mad Programmer to address the problems with out having actually seen them sometimes. We shared a lot of frustrations getting this far

Here are the results of many hours of design, testing, rewriting, and of course, lots of debugging

The I.S.T.U.G. Timex-Board has run online and 99% worry free for some time now, running with these features.

You do not need any expansion memory devices
A real time clock, advances the date at midnight
A 40 message message base
A Quick message scan that lists the mess. no., date and time left, and the To and From names
A users log, if you wish
A provision allowing some non-Timex callers to use our BBs. (some use a different code for NEWLINE/RETURN)
and the basic features of T/S Tinyboard
The callers name and date/time called, any messages to the SYSop,mess. read/left log, and when they logoff are all LPRINTED. It uses the 2040 printer for its ease, but any setup will do. Our SYSop has found it a great tool for managing the BBs.

Since none of us had an expanded machine, we used all the programming tricks we could (at least the ones I know) so that we could leave some free memory to allow changes for a mass storage device and its operation. Careful study of the program will show you that we did a lot of seemingly strange things with Basic. There is a reason to this madness. We worked real hard to save as much memory as we could and maintain reasonable speed. Where program speed is important we did ok. Where we wanted to control the speed or where memory saving tricks would not hurt us we did the best we could. CAUTION: Because parts of the program jump around be real careful if you start to change parts of the program we do not ask you to change.

FIRST make a copy of this program and set aside the original, work only on your copy. This is for your own good. Soon enough you are going to want to make changes and at least you will have something to fall back on when and if you mess up the program.

Now, lets decide what you are going to call your BBs.

With these instructions handy, lets set up the BBs.

All sections are identified with REM lines in INVERSE VIDEO so we are going to use the REM lines as locators for each segment as we proceed through the program setup.

Lets look at the idle screen first. You have the name for your BBs handy? Good, look for REM idle SCREEN$. Here you will find "The I.S.T.U.G. BBS", thats what you are going to change. You did leave the COPYRIGHT information alone didn't you. Bleave me, in it's current format it is important. Later you will see why.

Try this, find the REM logon DATA line. Direct command: RESTORE xxxx, xxxx=that line. Direct command again, READ A: FOR A=1 TO A: READ P$:PRINT P$:NEXT A. You can use this method to test your setups also. During all the debugging I used a PRINT GRAPHIC 8 after the NEXT A to see where the routine placed the last line.

Thats how this part of the BBs looks to the caller. You can use this method for each DATA segment in the program.

Format your logon screen and a bulliten screen now. When you think you have something you like lets examine the program so you can install your information.

The following information applys to all the DATA statements in the program.

The number in the first DATA statement sets the amount of lines sent by the program to the caller. If you see, or want a blank line sent use ,"", to do that, as the sample shows. Make a copy of your screen when you are done formatting it. Now set each line up as DATA segments. Limit each segment to 31 charactors or less if you can. If you need a full line the program will cause an extra linefeed to seem to have happened. That can let you elimanate a ,"", if you had 1 there. The total lines in your screen that you want the caller to see is the number you use in the DATA statement. We provide anextra DATA line to make it easy to format your version.

Meanwhile back at the REM logon DATA segment, there's our name again. Change it to yours. Next find REM menu DATA, and change the name there.
Now look at the REM bull DATA. We left a basic message here. We use this area to let callers know about our user group meetings, or other information that doesn't fit in the logon section. Did you change the name of the SYSop? Good for you. Please leave the Mad Programmers name intact, you don't want him to haunt you forever do you?

Next lets change the REM quit DATA. Here again you need to set up your closing message/comment. SPECIAL CAUTION: the line counter number MUST be 1 less then the total count. That's because the program uses the "Goodby, Hanging up now" statement later in the program. It works well, so pay heed. If you don't then you might mess up the program and you don't want to do that now do you? The Mad Programmer wants you to have a smooth running BBs.

Shall we discuss the users DATA next? Well I will. This is your BBs so we left the section in but empty. We list the frequent callers here. HINT: As you identify those you wish to place on your list keep track of the DATA statement count. You will use less memory if you list the long names first. We use this format DATA "Paul Holmgren   Willie Jones". Thus printing out 2 columns.
If you want to remove this feature, you need to change the REM menu DATA statements, delete the REM USR DATA area, and remove the U option in the menu select prompt line and the IF c$="u" OR c$="U" line.

One last change. Find REM edit. In this area find LET m$(33 TO)="SYSop Wille". Your SYSop's name goes here also.

If you have not gotten lost yet we are ready to RUN the program.

So, have you done that yet? Since this is the first time through you have no message base so answer the prompt with "n". Whoa, whats this? well, you are in the SYSop operations area. Get your watch out. This is next. Select E to establish correct time. Answer the prompts in the formats shown please. We are using a 24 hour clock, so time runs from 00 to 23 hours. All done? Lets check the time. Nice isn't it.

NOTE: At the end of the month please check and update the date information.

If you want to leave a message for new callers, do it now. We used message 1 to do that. Watch the prompts. After the message header PRINTs at the top of the screen you are ready to type. Here we used the INPUT function. Therefore typing a line of 31 charactors should be easy. At the end of each line INPUTed the program places a NEWLINE code. Corrections are done in the normal mannor. When you are done press ENTER, then SYMBL/SHIFT D= STEP, then ENTER again. The message is finished. If you use M or H at the SYSop menu, the program will be ready for it's first caller. GOOD LUCK!

Area by Area explanations of Use/Abuse/Highpoints.

Once again we are going to use the INVERSE VIDEO REM line locators for each segment.

First: Error trapping is for when something happens that ended up stopping/crashing/locking up the host computer. We did not want the operations of the program or any inputs by the caller or even the caller hanging up at odd times to tie up the computer.
It is used all over the program. There are 2 main methods.
1 using ON ERR THEN GO TO XXXX
2 testing for certian conditions

Early on we used GO SUB/RETURN a lot, but that proved to be a major contributor to crashing. A way was found around that. The variable ret=RETURN, ergo LET ret=xxxx/GO TO yyyy/GO TO ret handles all that now.

With that out of the way lets concider REM BEEP: the loop resets x to =1, After it's use. We use x as 1 several times. You can change the BEEP but please leave the loop alone.

REM test FOR sysop access: Here again the 7 bit status is confirmed. If IN<128 tests for an active modem connection. We also test for an interuption request by the SYSop for CHAT or EDIT mode. SYMBL/SHIFT D= STEP allows you to go to SYSop edit mode from several points. Holding C down will take you to CHAT mode.

REM start/set on LINE time: Here we allow those calling that cann't use a linefeed control code= 13 to use the BBs and set a variable to time the call. We also call a routine, if needed, that changes all the embedded codes in the message base to match.

REM menu: of course has to handle upper/lower case.

REM CLS SCREEN$,12=FF: At this time a form feed works great to clear the callers screen.

REM FORMAT & READ header: The string slicing also controls the timing of the printout to the caller. We think it works great without any PAUSE.

REM READ mess: Only numbers are allowed here and the caller must use a RETURN/NEWLINE/ENTER to proceed. We don't let them read a mess. that is not there. For the SYSop the screen clears then says "Working", that way you know something is going on. Of course we only send the caller the valid length of the message. CONTROL C=3 is the flag for that operation. Doing it that way speeded up the output to the caller. Then the printer makes noises. (Prints out "Read mess. #x)

REM con=CONTINUE: Here we give the caller control over when to proceed for another action.

REM leave message: First we ask if the mess. is for the SYSop, if so then we set up to LPRINT the mess. and not putting it in the mess. base. More on that later. Lets find the first available blank mess.. Now that we have it the caller gets a chance to enter a mess.. All done?: Tell the caller "Mess. saved", if the caller fills the mess. then close the mess. and tell the caller "Mess. full and saved".

REM Quick scan: Prints mess. no., date and time mess. left, and the To:/From: info. The caller can use a CONTROL C to abort the scan.

REM chat: Seems normal until you hit ENTER, that causes the caller to get a ">" prompt on his screen. Likewise if the caller uses ENTER/NEWLINE then you get a flashing ">" on your screen. CAUTION: If you (as sysop) make a typing error DO NOT use DELETE. Use the LEFT ARROW. DELETE CLEARS the callers screen. Just count in your head how far to go back and use the LEFT ARROW to do that. Hitting any key when the caller is requesting CHAT should take you there. SYMBL/ SHIFT D= STEP will return you to the SYSop menu.

REM quit BBs: Here we allow the caller to RETURN to the main menu if he got here by mistake. If the caller wishes to leave then we use the last DATA segment from the quit DATA area, and send the callers date, time of day, and time online to the caller and your printer.

REM timer: The first line does the major time computations. If you find your computer is not keeping accurate time change the 3600 in the first calculation in steps of 5 first. The rest of the routine converts the calculations into a usable time representation, both for current local time and for the callers online time.

REM edit: First we send a mess. to the caller that we are taking control of the program, then you are in EDIT mode. Use E to correct the clock and to set the time before you use the program. Use H or M to leave EDIT and to return the caller to the program. If you access EDIT during an idle time this also resets the program to Scanning Phone Line because the error trapping has tested for a modem in use and decided that since no one is there lets disconnect and wait for a call. Use L, or Q and the program uses a variable flag so that you can use a part of the program for each request's use without having separate sections doing the same work. The program prints to your screen and not to over the modem. Here also we used a prompt allowing control of the programs proceedings. IMPORTANT: Every time you use the SAVE or LOAD function you will have to E stablish the correct time again.

REM error trapped messages: Here we placed messages to indecate to the proper party a trouble condition occured.

REM LPRINT SYSop mess: Even tho we have a 40 message setup we DIMed 41, m$(41) is used to temporarily store the mess. for the SYSop. So the mess. no. is set to 41 and then jump to regular leave mess. section to fill the mess. and return here to LPRINT it.

REM init: of course you know what this area is for. SEVERAL things to watch for.
sm = 41, this is done to save memory later, it's used elsewhere and it works. Dodn't lose it.
this is where d$ is setup. Neat trick: if you save the mess. base we LET m$(sm)=d$, this stores the DATE/TIME info for the messages with the mess. base in a single SAVE. When you LOAD the mess. base back in we then LET d$=m$(sm). Voila! through the miracle of string slicing the messages are now date/time marked again when it's needed.
CAUTION: The machine code for allowing other callers using a different control code for linefeeds is designed for m$(41,600), thats 24,600 bytes it searches, don't change it.

REM SAVE: If you are going to save the program and use an existing mess. base later, save the mess. base using the EDIT mode first. Then you can BREAK the program and RUN this line.

If you start to add/make changes and end up hitting ERROR REPORT stating out of memory, you can DELETE the REM lines, they use over 600 BYTES of memory. Also CLEAR 65535 gets you 168 bytes more.

If you want to change the size of the mess. base the machine code will have to be changed. If there are any changes you wish to make and the programming is a bother, write and tell me what you are trying to do. There is a real good chance we can help you get just what you want within the confines of the programs operations.

The version of this program I.S.T.U.G. is using has been modified to the SYSop's liking. As you better understand the program you can make changes that you want.

Paul Holmgren [redacted]
Indy, In.

The variables used, and why they are named thusly

rcv	receive, gets information from the caller
xmt	transmit, primes modem to send info to caller
t$(2,5)	time string, t$(1) stores callers on line time, t$(2) stores current time
temp	base time mark for callers time on line
dy, mo, yr	day, month and year
min, hr	current miniute and hour
time	used in computing the time settings

T$, hr, min are reset each time the time subroutine is accessed

mc	the port we test for an active modem connection
nl	routine to line feed you and the caller
sm	routine to send message in P$ to caller, variable is also used at this time in some routines to save memory
P$	contains any information to be sent to the caller
beep	routine to signal Sysop and reset the x variable sometimes
menu	main menu line address for the caller
cls	clear and reset the callers screen
con	a prompt for the caller to control a pause
l$	this is the line feed value sent to the caller
d$(600)	holds the date and time data for the messages in the message base, it gets string sliced alot
m$(41,600)	the message base
c$ and c	convert callers input to something the program can use
x and y	FOR / NEXT loop variables
P1 nd P2	position variables in the Quick Scan routine
ed	flag used in the Sysop edit menu operation
P3	another flag, used in leaving a less then full message to the sysop
xmt1/rcv1	used in chat for charactor transmit/receive
n$	name string holds callers name

	Key	NAME		MTERM function
00		NUL	null	
01		SOH	start of heading	
02		STX	start of text	
03		ETX	end of text	
04		EOT	end of transmission	
05		ENQ	enquiry	
06		ACK	acknowledge	
07	G	BEL	bell	ring bell
08	H	BS	back space	curser back 1 space and erase
09		HT	horizonal tab	curser right to next 8 tabposotion
10	J	LF	line feed	curser down and start of new line
11		VT	vertical tab	
12	F	FF	form feed	scroll screen (8 ? lines)
13	M	CR	carriage return	curser down and start of new line
14	N	SO	shift out	curser on
15	O	SI	shift in	curser off
16		DLE	data link escape	
17		DC	device control	
21		NAK	negitive acknowledge	
22		SYN	synchronous idle	
23		ETB	end of trans. block	
24	X	CAN	cancel	curser left on column
25	Y	EM	end of medium	" right 1 "
26	Z	SUB	substitute	" down 1 line
27		ESC	escape	curser up 1 line
28	2	FS	file separator	curser home, top line
29	3	GS	group "	return curser to start of line
30	4	RS	record "	erase to end of line
31	5	US	unit "	" " " " screen
32		SP	space (blank)	
127		DEL	delete	

CAPS SHIFT = 12 clears the callers screen

OUT = 0 hang up
1 end
2 pulse kill tone but not disconnect
3 starts
4 pulse

31 inits phone for auto dial

34 starts carrier tone

133 carrier tone present

```
ADDR HEXCODE LABEL MNEMONIC
===========================
685B AF      26715  XOR A          CHAT  LET A = 0
685C DB77           IN A,(77)            LET A = PEEK 119
685E E603           AND 03                A = 0 OR 3 ONLY
6860 0600           LD B,00              LET B=00
6862 4F             LD C,A               LET C= A
6863 C9             RET                  RETURN
6864 AF      26724  XOR A          RCV   LET A=0
6865 DB77           IN A,(77)            LET A = PEEK 119
6867 E680           AND 80               IF A>128 THEN LET A = 128
6869 C8             RET Z                IF Z = 0 THEN RETURN
686A AF             XOR A                LET A = 0
686B DB77           IN A,(77)            LET A = PEEK 119
686D E602           AND 02                A = 0 OR 2 ONLY
686F 28F4           JR Z,6865            IF Z=0 THEN GOTU 6865
6871 DB73           IN A,(73)            LET A = PEEK 115
6873 010000         LD BC,0000           LET BC=0000
6876 4F             LD C,A               LET C= A
6877 C6E8           ADD. A,E8            LET A = A+232
6879 C60A           ADD A,0A             LET A = A+10
687B 38F4           JR C,6871            IF C=1 THEN GOTO 6871
687D DB77           IN A,(77)            LET A = PEEK 119
687F E680           AND 80               LET A = 0 OR 128 ONLY
ADDR HEXCODE LABEL MNEMONIC
===========================
6881 C8             RET Z                IF 2=0 THEN RETURN
6882 AF             XOR A                LET A=0
6883 DB77           IN A,(77)            LET A = PEEK 110
6885 E601           AND 01               LET A = 0 OR 1 ONLY
6887 28F4           JR Z,687D            IF 2=0 THEN GOTO 687D
6889 79             LD A,C               LET A = C
688A D373           OUT (73),A           POKE 115, A : RETURN
688C C9             RET
688D 00      26765  NOP                  XMT
688E AF             XOR A
688F DB77           IN A,(77)
6891 E680           AND 80
6893 C8             RET Z
6894 AF             XOR A
6895 DB77           IN A,(77)
6897 E601           AND 01
6899 28F4           JR Z,688F
689B 18C1           JR 685E
689D 00             NOP            INIT MODEM
689E 3E02    26782  LD A,02
68A0 D377           OUT (77),A
68A2 3E22           LD A,22
ADDR HEXCODE LABEL MNEMONIC
===========================
68A4 D377           OUT (77),A
68A6 3E40           LD A,40
68A8 D377           OUT (77),A
68AA 3E7B           LD A,7B
68AC D377           OUT (77),A
68AE 3E37           LD A,37
68B0 D377           OUT (77),A
68B2 C9             RET


[illegible]  [illegible]0
6[illegible]
6[illegible] 4[illegible]
68BC 2053
68BE 4F
68BF 46
6[illegible] 54
6[illegible] 57
```

MULTI BASE